

Explain Plan

# Explain plan příkaz v konzoli

- Nejprve vykonejte samotný SELECT, který bude začínat s EXPLAIN PLAN FOR:

```
EXPLAIN PLAN FOR
```

```
SELECT * FROM hr.employees
```

```
WHERE job_id = 'IT_PROG' AND salary < 5000;
```

- Poté můžete získat statistiky tímto způsobem:

```
SELECT * FROM
```

```
table(dbms_xplan.display(NULL, NULL, 'typical'));
```

# Explain Plan vs. Autotrace

- Jaký je rozdíl mezi těmito tlačítky v SQL Developeru?
  - **Explain Plan** zobrazuje jak bude Oracle optimizátor procházet SELECT (bez samotného vykonání SELECTu). Jedná se o předpoklad jak bude SELECT vykonáván.
  - **Autotrace** používá plán a vykonává samotný SELECT. Výsledek Autotrace je tedy skutečný výsledek běhu SELECTu.
  - [http://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11\\_QUESTION\\_ID:296280200346630999](http://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:296280200346630999)

# Autotrace SELECT

- Pro tento SELECT vrací Autotrace následující výsledek:

```
SELECT * FROM hr.employees
```

```
WHERE job_id = 'IT_PROG' AND salary < 5000;
```

„Cena“ dotazu – čím větší, tím horší.  
Zahrnuje CPU, IO a zatížení přenosové sítě

Kolik bloků musel  
Oracle přečíst.  
Čím větší, tím horší.

OPERATION	OBJECT_NAME	CARDINALITY	COST	LAST_CR_BUFFER_GETS
SELECT STATEMENT			2	
TABLE ACCESS (BY INDEX ROWID)	EMPLOYEES	1	2	2
Filter Predicates SALARY<5000				
INDEX (RANGE SCAN)	EMP_JOB_IX	5	1	1
Access Predicates JOB_ID='IT_PROG'				

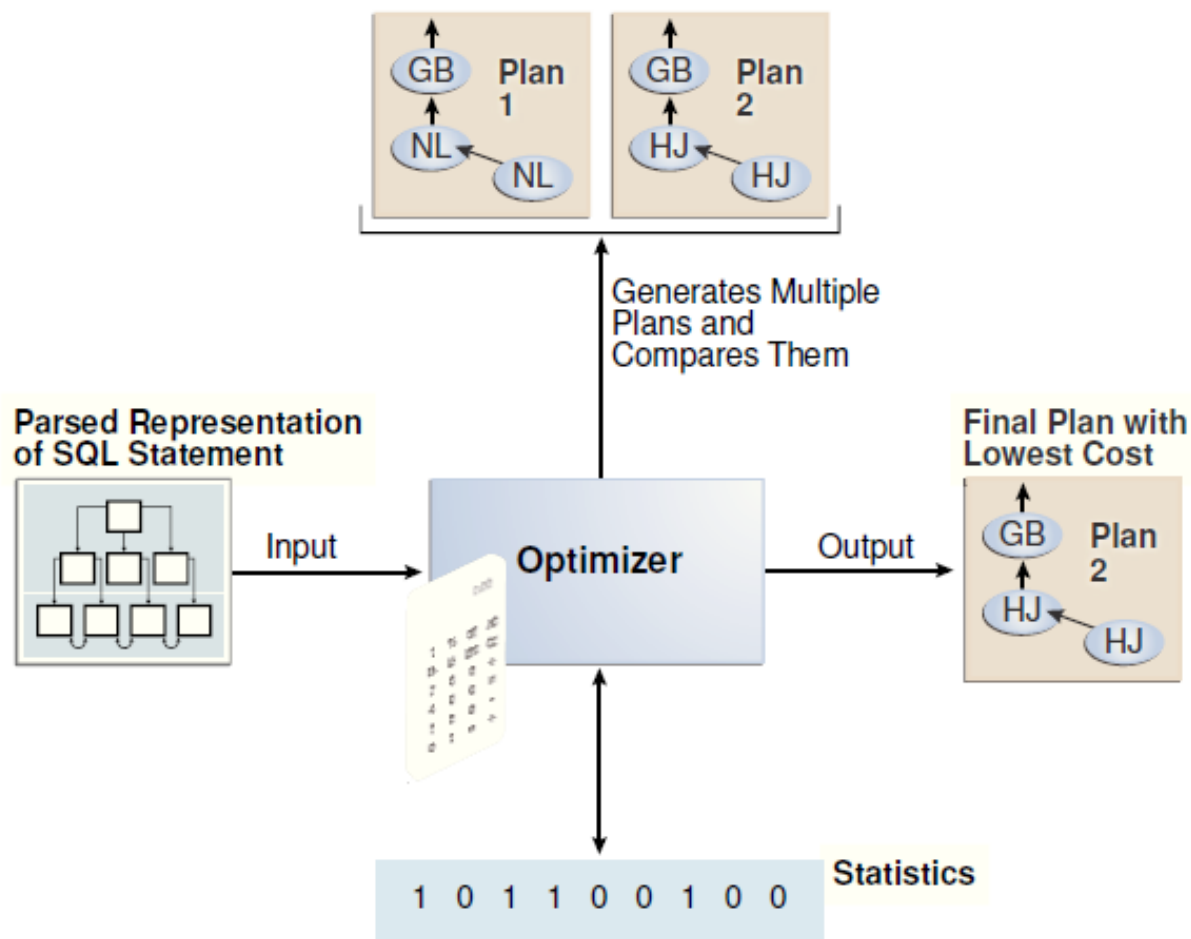
Jednotlivé operace,  
ze kterých se skládá SELECT.

Názvy použitých  
DB objektů

Předpokládaný počet řádků,  
které budou výsledkem operace.

# Základní pojmy

- Explain plan je vítězný Execution plan. Oracle Optimizátor parsne vstupní SQL dotaz, sestaví sérii exekučních plánů (SEQ scan, index 1, index 2, ...) a vybere ten s nejmenší cenou:



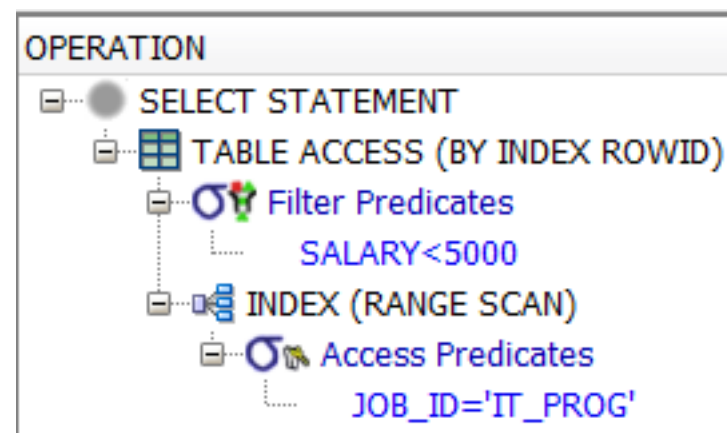
# Běh Explain Plan

- Obyčejně se Explain Plan interpretuje následovně:
- Tento Explain Plan má tři operace:

(1) SELECT STATEMENT

(2) TABLE ACCESS ...

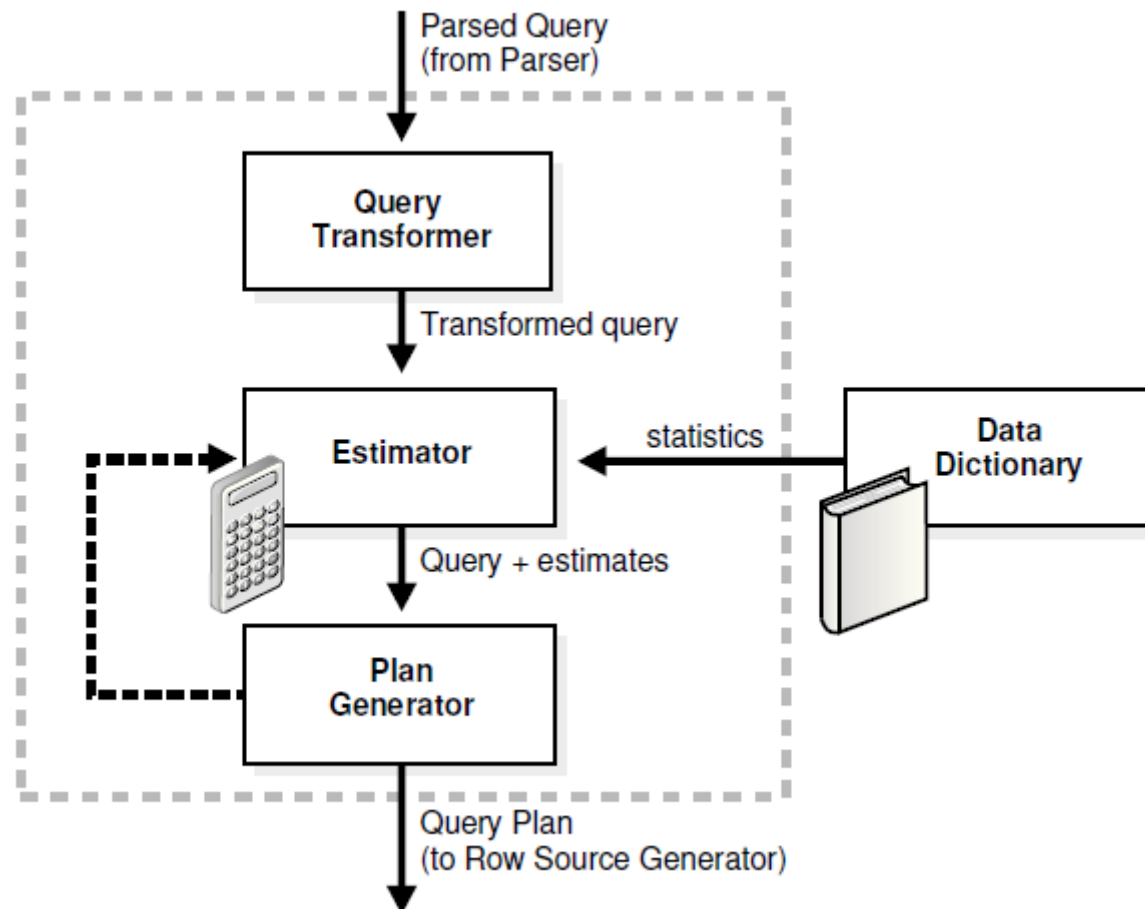
(3) INDEX ...



- Tyto operace se vykonají v následujícím pořadí: (3), (2), (1).
- Vždy se vykoná nejvnitřnější operace, jejíž výsledek se použije nadřazenou operací. Každá z operací se obvykle vykoná právě jednou.
- **Poznámka:** Toad zobrazuje pořadí operací!!!

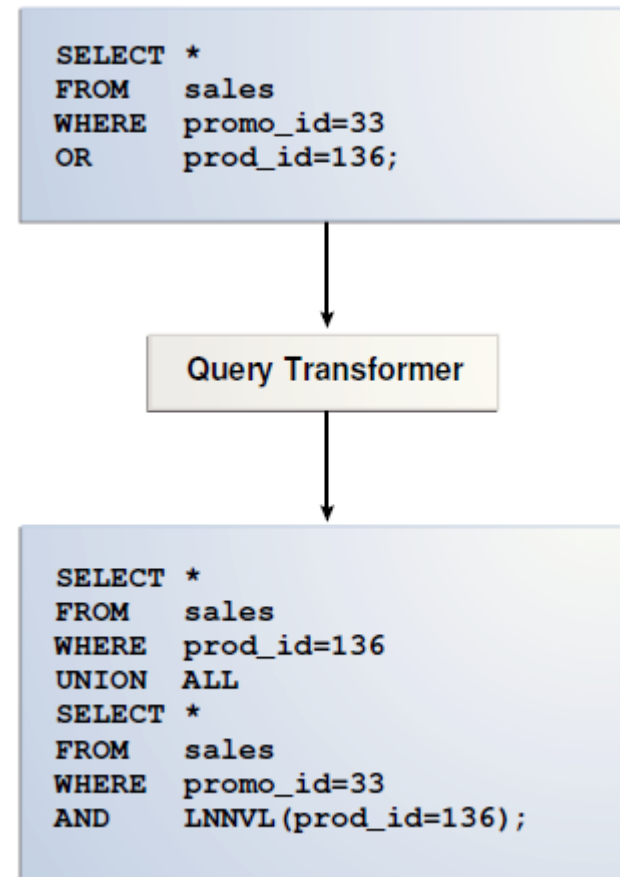
# Oracle Optimizer

- Oracle Optimizer se skládá z následujících komponent:



# Query Transformer

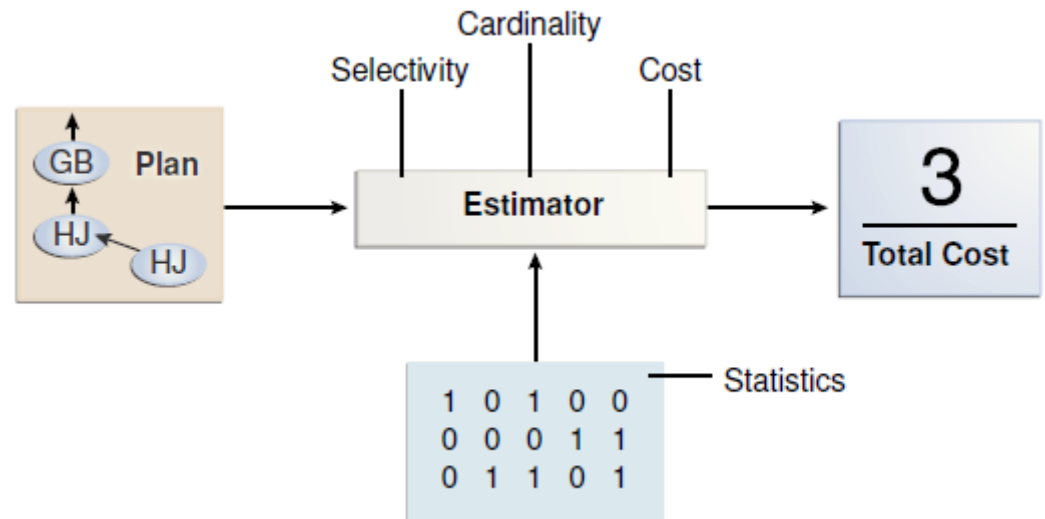
- Query Transformer rozhoduje o tom, jestli přepíše původní SQL dotaz do sémanticky ekvivalentního SQL dotazu s nižší cenou.





# Estimator

- Estimator počítá výslednou cenu každého exekučního plánu. K tomu používá tři hodnoty:
  - Selectivity – procento řádků, které dotaz vybírá
  - Cardinality – počet řádků vrácených každou operací v exekučním plánu
  - Cost – cena prostředků (I/O, CPU, memory)

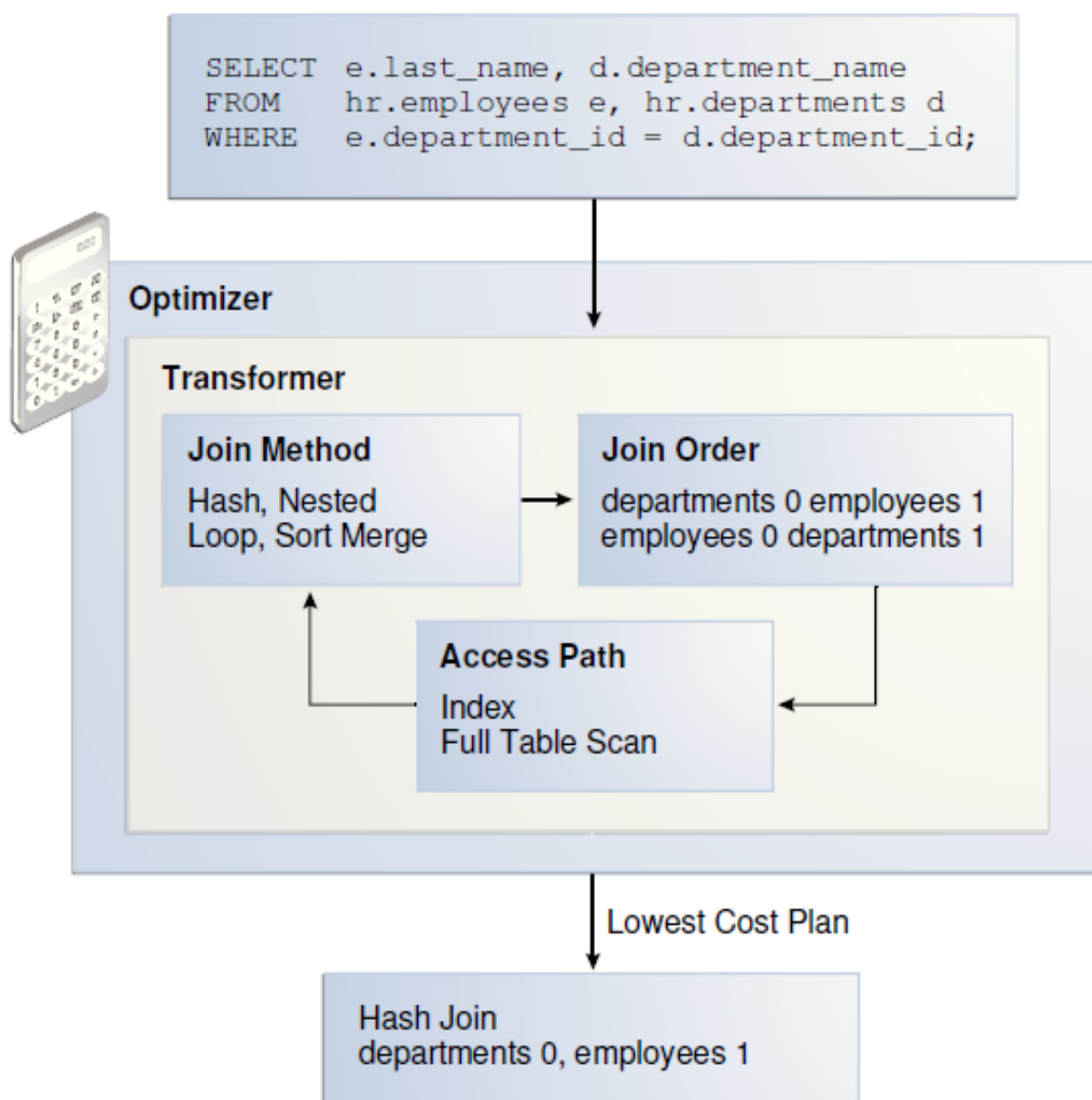


# Statistiky

- Statistiky jsou velice důležité, protože je Estimator používá pro spočítání selectivity, cardinality i cost.
- Když statistiky nejsou k dispozici, pak se například snaží Oracle uhádnout selectivitu podle použitého operátoru (například operátor = (rovnost) pravděpodobně vrátí menší množství záznamů než operátor > (větší)).

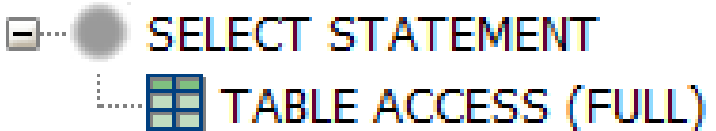
# Plan Generator

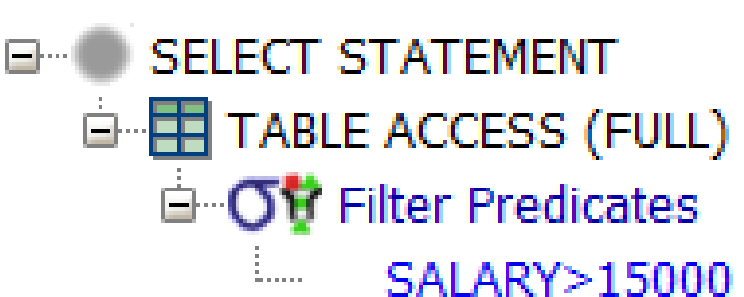
- Plan Generator zkouší různé typy přístupů (SEQ., index 0, index 1, ...) a vybírá exekuční plán s nejmenší cenou.



# TABLE ACCESS (FULL)

- **Full Table Scan:** Přečte všechny záznamy z tabulky sekvenčním způsobem.
- Tento způsob se vybere:
  - Když se má projít hodně záznamů z tabulky.
  - Když se pro přístup nepoužívají indexy.
  - Když použití indexu má větší cost než sekvenční přístup.
- Příklady:

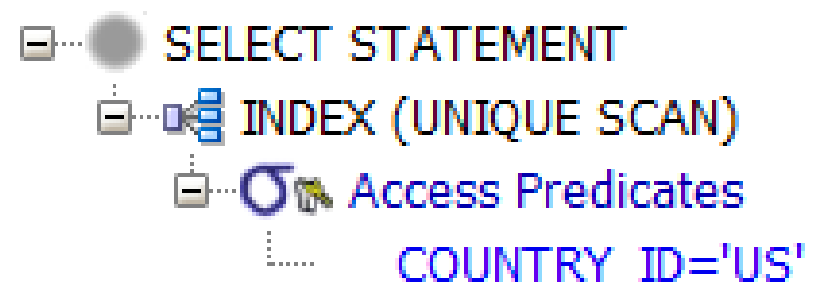
`select * from hr.employees;` 

`select * from hr.employees  
where salary > 15000;` 

# INDEX (UNIQUE SCAN)

- **Index Unique Scan:** Když je výsledkem SELECTu jeden řádek získaný pomocí jednoho unique indexu.
- Příklad:

```
select * from hr.countries  
where country_id = 'US';
```



# INDEX (RANGE SCAN)

- **Index Range Scan:** Když je výsledkem SELECTu více řádků na základě unique indexu, nebo když je kritérium postavené na základě non-unique indexu.

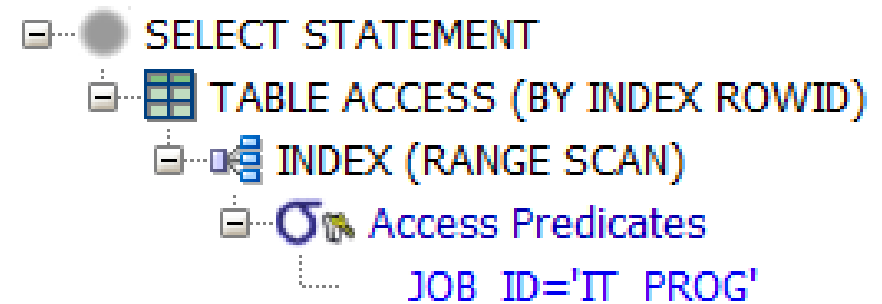
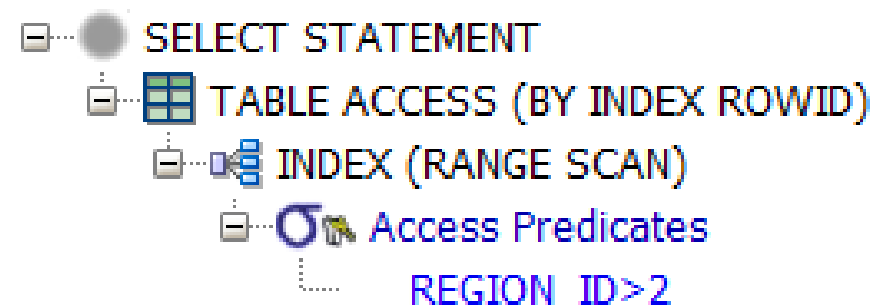
- Příklady:

```
select * from hr.regions  
where region_id > 2;
```

```
select * from hr.employees  
where job_id = 'IT_PROG';
```

- Pozor! Potenciálně pomalé!

(Záleží na tom, kolik záznamů se tímto způsobem vrací)



# Index s více sloupci nebo více indexů nad jedním sloupcem?

- Velice častá otázka. Odpověď: Záleží :-)
  - <http://stackoverflow.com/questions/179085/multiple-indexes-vs-multi-column-indexes>
- Každopádně pokud máme v SELECTu ve WHERE podmínce omezení podle dvou sloupců, pak je efektivnější mít jeden složený index.
- Nicméně od Oracle 8 se v rámci jedné podmínky může použít více indexů.
- Praktický příklad je na další stránce.

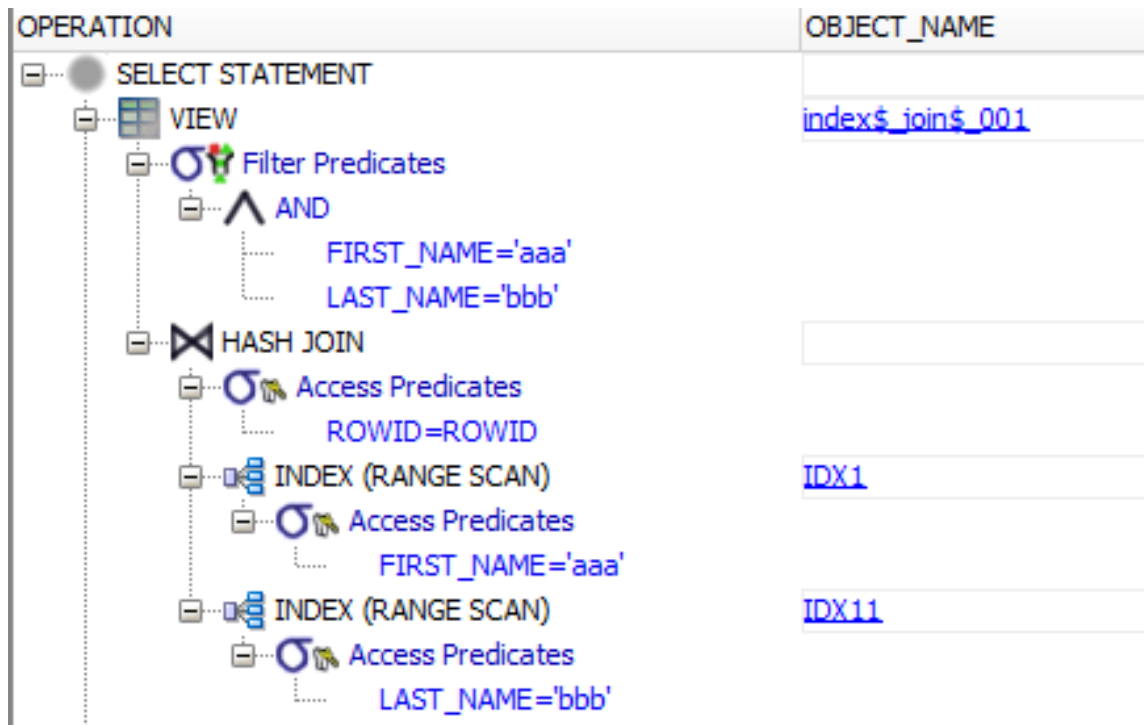
# Index merge – příklad

```
Select * from text_data
```

```
where first_name = 'aaa' and last_name = 'bbb';
```

```
create index idx1 on text_data (first_name);
```

```
create index idx11 on text_data (last_name);
```





# INDEX (SKIP SCAN)

- U indexů, které mají více sloupců je důležité jejich pořadí (index je možné použít pouze když výraz ve WHERE podmínce obsahuje sloupce ze začátku indexu).
  - Příklad: index se sloupci (A, B, C)
  - Aby se index použil, je nutné mít ve WHERE podmínce omezení podle sloupců (A, B, C), (A, B), (A), (A, C)
  - Pro omezení (B, C), (B), (C) se index nepoužije.
    - Od Oracle 9 se v některých situacích použije INDEX SKIP SCAN.
    - <https://oracle-base.com/articles/9i/index-skip-scanning>

Příklad na nutné pořadí: test-multi-value-index.sql

# Pořadí sloupců v multi-column indexu

- Co nejvíc selectivní sloupec by měl být dřív. Příklad:
  - Sloupce: FIRST\_NAME, LAST\_NAME
  - Ve FIRST\_NAME jsou stovky unikátních hodnot
  - V LAST\_NAME jsou tisíce unikátní hodnot
  - Kompozitní index by měl tudíž být:
    - (LAST\_NAME, FIRST\_NAME)
- Na druhou stranu v moderních databázích nezáleží na pořadí sloupců ve WHERE podmínce.

# Use The Index, Luke!

- <http://use-the-index-luke.com/sql/preface>

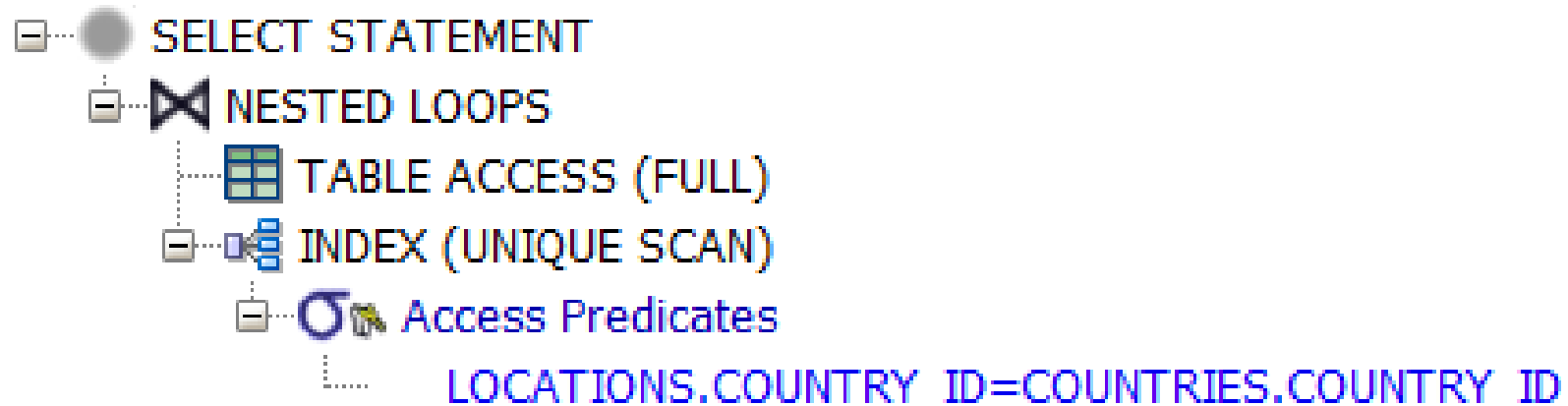
# SORT

- Při třídění pomocí klauzule ORDER BY se také bere v úvahu jestli je na sloupcích, pomocí kterých se třídí nastaven index, ale tady hodně záleží na optimalizátoru jestli ho použije nebo ne.
- Zejména pokud se třídí podle více kritérií nebo se vrací hodně záznamů, pak Oracle preferuje sekvenční přístup.

# NESTED LOOPS

- Při spojování tabulek může použít Oracle optimalizátor několik přístupů.

```
select * from hr.locations join hr.countries  
using (country_id)
```

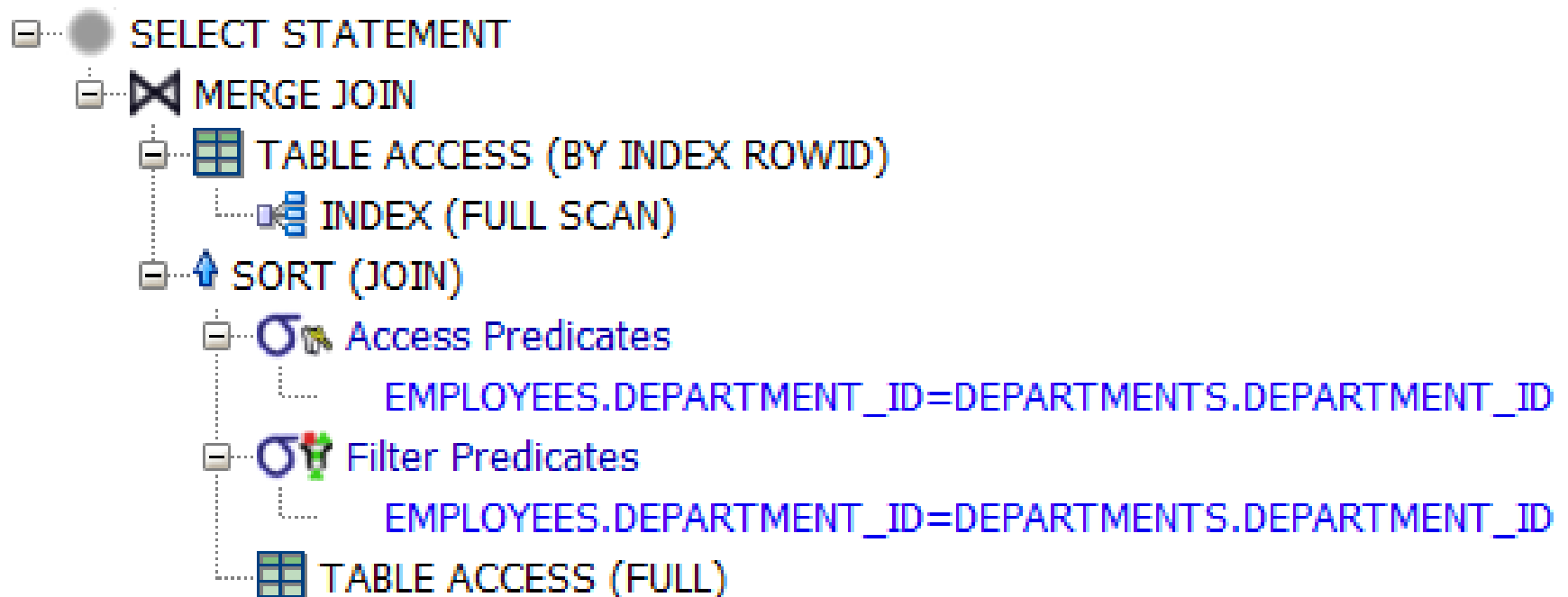


- **Nested loops** se použije, když se spojují tabulky s malým množstvím dat a data se spojují přes primární a cizí klíč. Fyzicky se pro každý řádek první tabulky provede FOREACH cyklus v druhé tabulce.

# MERGE JOIN

- V případě, že spojovací podmínka není rovnost (=), ale jeden z operátorů: >, <, >=, <= pak se může použít MERGE JOIN.
- Také se použije, když je v jedné z tabulek použit primární klíč:

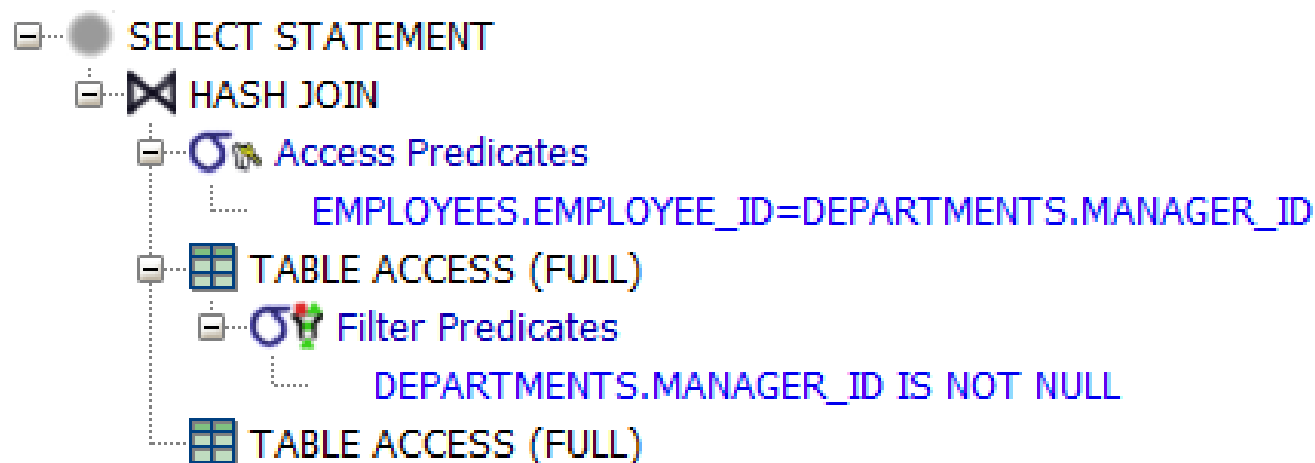
```
select * from hr.employees join hr.departments  
using(department_id);
```



# HASH JOIN

- Hash join se použije při spojování tabulek (nebo obecně matic) s velkým množstvím záznamů, nebo když se pro spojení tabulek nepoužijí indexy. Optimalizátor vezme menší ze dvou tabulek, vytvoří v operační paměti hash tabulku (na základě klíče použitého pro spojení obou tabulek) a poté prochází větší tabulku, na její klíč aplikuje stejný algoritmus a zjišťuje, jestli odpovídá záznamu v hash tabulce. Pokud ano, pak vrátí záznam.

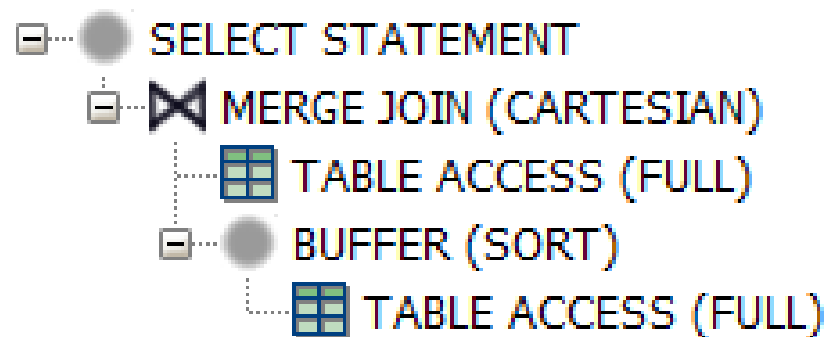
```
select * from hr.employees join hr.departments  
on employees.employee_id = departments.manager_id;
```



# CARTESIAN JOIN

- Nejvíce náročné spojení je kartézský součin, velice často se jedná o chybu v dotazu.
- Příklad:

```
select * from hr.employees, hr.departments;
```

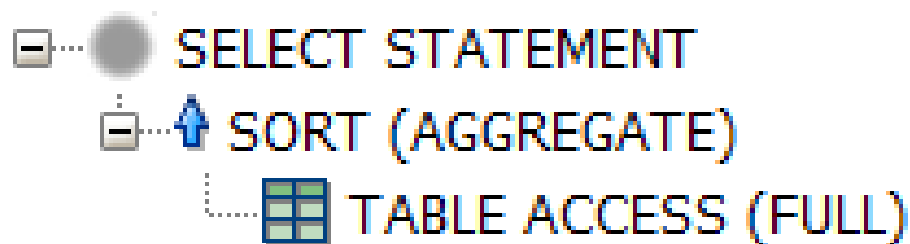




# Agregační funkce

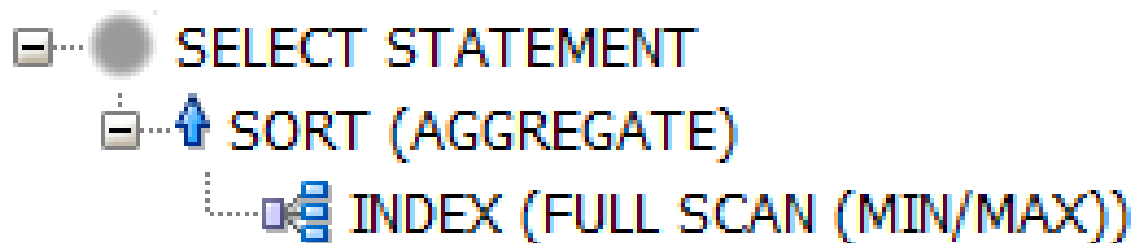
- Použití agregační funkce vypadá následovně:

```
select max(salary) from hr.employees;
```



- Agregační funkce na sloupci, na kterém je nastaven index vypadá takto:

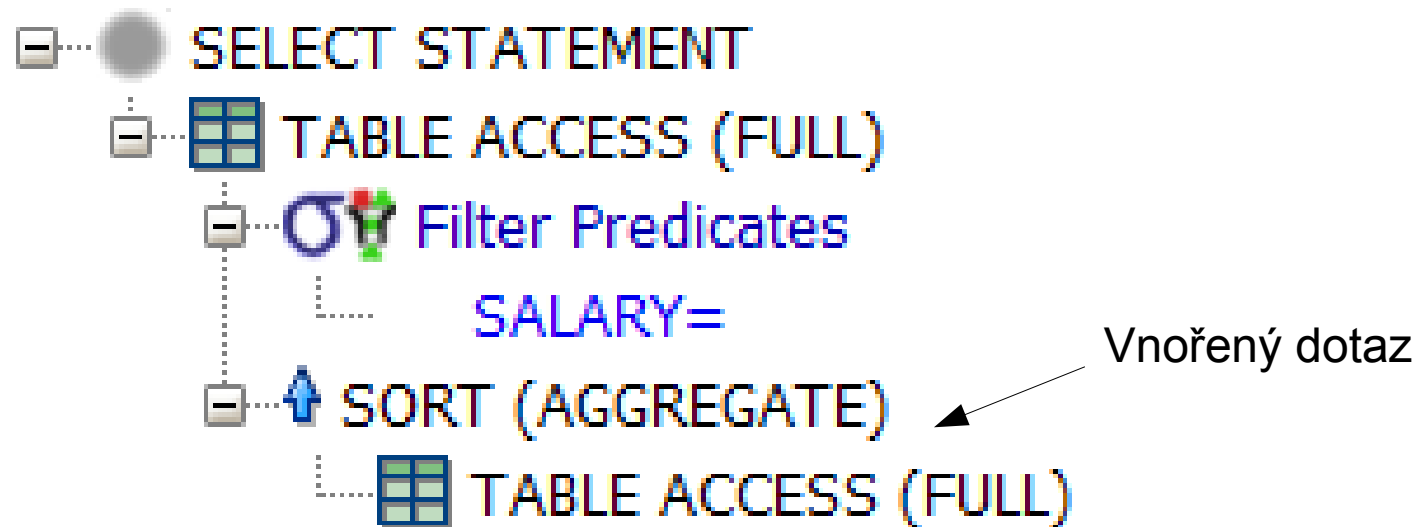
```
select min(employee_id) from hr.employees;
```



# Vnořené dotazy

- Vnořené dotazy se vykonávají následovně:

```
select * from hr.employees where salary =  
(select max(salary) from hr.employees);
```

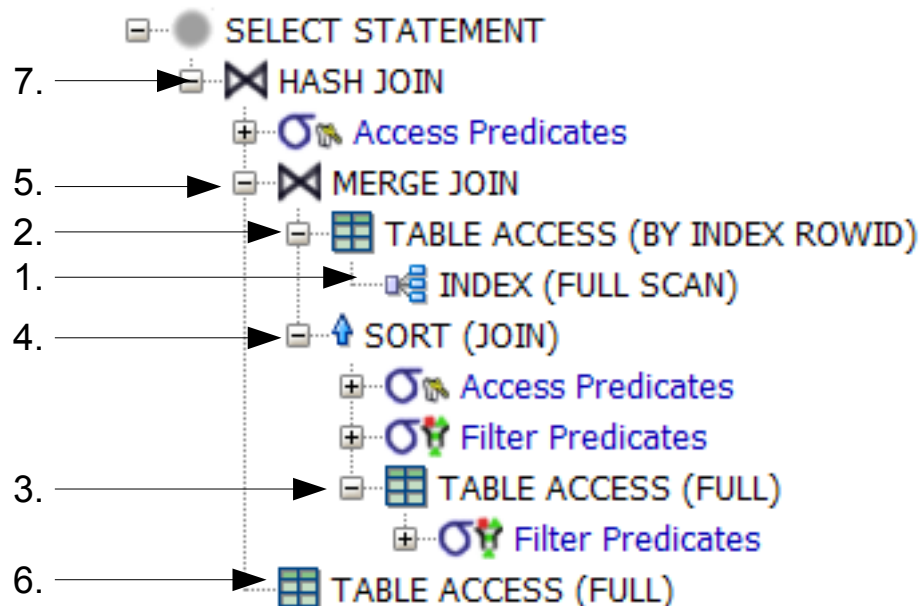


- Nejprve se vykoná vnořený dotaz, jehož výsledek se použije v nadřazeném dotazu.

# Čtení pokročilých plánů

- Obvykle nejvíce vnořená operace se vykoná jako první. Pokud je na stejné úrovni hierarchie více operací, pak se obvykle vykonávají v sekvenčním pořadí:

```
select * from hr.employees join hr.jobs using (job_id)
join hr.departments using (department_id)
where salary > 10000;
```



## Tipy:

- Skryjte Access & Filter Predicates
- Vše ostatní jsou operace
- Obvykle se začíná operací, která má nejmenší „cost“
- Toad for Oracle zobrazuje pořadí vykonávaných operací (ale nezobrazuje Predicates a je placený)

# Literatura

- <http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-explain-the-explain-plan-052011-393674.pdf>
- [http://www.dba-oracle.com/t\\_order\\_sequence\\_sql\\_execution\\_explain\\_plans\\_steps.htm](http://www.dba-oracle.com/t_order_sequence_sql_execution_explain_plans_steps.htm)